REMARKS

Applicant respectfully requests the Examiner's reconsideration of the present application. No claims have been cancelled. Claims 12, 16, 18 and 20 have been amended. No claims have been added. Therefore, claims 12-22 are presented for examination.

Claim Objections

Claims 12, 16, 18, and 20 are objected to because of informalities. Applicant has amended claims 12, 16, 18, and 20 to address the informalities, and respectfully submits that the amendments overcome the Examiner's objections. Applicant submits that no new matter has been added by these amendments. Accordingly, Applicant requests that the objections to the claims be withdrawn.

Rejections Under 35 U.S.C. §103(a)

Blodget

Claims 12-22 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent Number 6,510,546 by Blodget (hereinafter "Blodget"). Blodget qualifies as prior art only under 35 U.S.C. § 102(e) because its filing date of July 13, 2000 is earlier than Applicant's filing date of August 8, 2000, and its issue date of January 21, 2003 is later than Applicant's filing date. Applicant is submitting herewith an affidavit under 37 C.F.R. § 1.131 that the invention as claimed in claims 12-22 was reduced to practice prior to the July 13, 2000 filing date of Blodget. Therefore, Blodget is not available as prior art under 35 U.S.C. § 102(e), and Applicant accordingly requests that Blodget be removed from consideration. Accordingly, Applicant respectfully requests the withdrawal of the rejection of claims 12-22 under 35 U.S.C. § 103(a).

Conclusion

Applicant respectfully submits that in view of the amendments and discussion set forth herein, the applicable rejections have been overcome and the pending claims are in condition for allowance.

If the Examiner determines the prompt allowance of the claims could be facilitated by a telephone conference, the Examiner is invited to contact Tom Ferrill at (408) 720-8300.

Authorization is hereby given to charge our Deposit Account No. 02-2666 for any charges that may be due.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: 8/17, 2004

Jeffery Scott Heileson
Reg. No. 46,765

12400 Wilshire Blvd.
Seventh Floor
Los Angeles, CA 90025
(408) 720-8300

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	Examiner: Thompson, Annette M.
Drew Eric Wingard et al.	Art Group: 2825
Application No.: 09/634,045	Confirmation No.: 5608
Filed: August 8, 2000	
For: LOGIC SYSTEM WITH CONFIGURABLE INTERFACE	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION UNDER 37 C.F.R. § 1.131

We, Drew Eric Wingard, Michael J. Meyer, Geert Rosseel, Lisa A. Robinson, and Jay S. Tomlinson, declare the following:

1. We are the inventors of the above identified patent application. We are also employees or former employees of the assignee, Sonics, Inc. (hereinafter "Sonics"), of the application.

2. We have reviewed the application, including the claims of the application, and we have also reviewed a copy of the current claims which are pending (a copy of which is attached as Exhibit A).

3. The declaration made herein is to establish reduction to practice prior to **July 13, 2000**, which is the effective filing date of U.S. Patent 6,510,546 to Blodget.

4. We reduced to practice the claimed invention (in the claims of Exhibit A) prior to July 13, 2000.

5. Exhibit B attached herewith is an Invoice from Sonics to VxTel, Inc. (hereinafter "VxTel"), describing the sale of the Sonics SOC Integration Suite product prior to July 13, 2000. The SOC Integration Suite product included embodiments of the claimed invention. Exhibit B demonstrates that the claimed invention was reduced to practice prior to July 13, 2000. Note that the invoice date and purchase amounts have been redacted.

6. Exhibit C attached herewith is a Purchase Order from VxTel to Sonics, describing the purchase of the Sonics Integration Suite product prior to July 13, 2000. The SOC Integration Suite product included embodiments of the claimed invention. Exhibit C demonstrates that the claimed invention was reduced to practice prior to July 13, 2000. Note that the purchase order date and purchase amounts have been redacted.

7. Exhibit D attached herewith is product manual for the Sonics Integration Suite product, entitled SOCIntegrator and SOCBuilder User Reference. The product manual accompanied the sale of the Sonics Integration product described above, prior to July 13, 2000. Embodiments of the invention are supported throughout the entire product manual, but are more specifically described starting at page 32, and in general in Chapter 5 at page 43 of the manual. Exhibit D demonstrates that the claimed invention was reduced to practice prior to July 13, 2000.

8. Exhibit E attached herewith is a press release from EDN dated September 16, 1999, entitled Design Suite Accelerates SOC Development. The press release describes some aspects of the functionality of the Sonics SOC Integration Suite. Exhibit E demonstrates that the claimed invention was reduced to practice prior to July 13, 2000.

9. Exhibit F attached herewith is a press release from EE Times dated September 7, 1999, entitled Sonics Spins SoC Plug-and-play Tool. The press release describes some aspects of the functionality of the Sonics SOC Integration Suite. Exhibit F demonstrates that the claimed invention was reduced to practice prior to July 13, 2000.

10. Based on the above description and as is evident from the attached exhibits, reduction to practice of the subject matter, for its intended purpose, was accomplished at least prior to July 13, 2000.

11. We declare, to the best of our knowledge, that all statements made in this document are true, and that all statements made on the information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under § 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-identified patent application or any patent issued thereon.

Dated: 8/10/2004 — Name: Drew Eric Wingard

Dated: 8/11/2004 — Name: Michael J. Meyer

Dated: 8/11/2004 — Name: Geert Rosseel

Dated: 8/13/2004 — Name: Lisa A. Robinson

Dated: 8/11/2004 — Name: Jay S. Tomlinson

Attorney's Docket No. 2998.P011 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	Examiner: Thompson, Annette M.
Drew Eric Wingard et al.	Art Group: 2825
Application No.: 09/634,045	Confirmation No.: 5608
Filed: August 8, 2000	
For: LOGIC SYSTEM WITH CONFIGURABLE INTERFACE	

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

EXHIBIT A

The claims listed below are currently pending in the present Application, as presently amended:

12. A new computer core having an interface to communicate with other cores, wherein the interface contains a plurality of interface signal carriers that are configurable, at compilation, such that at least one of the interface signal carriers is selectively physically present in the interface or not physically present, wherein not physically present means that a route connection is not generated for an interface signal carrier selected to be not physically present.

13. The computer core as set forth in claim 12, wherein the computer core is a core on a system on a chip and the other cores also belong to that system on a chip.

14. The computer core as set forth in claim 12, wherein a first interface signal carrier is further configured to support different levels of functionality for the interface.

15. The computer core as set forth in claim 14, wherein a signal carrier width of the first interface signal carrier is also configurable to support different signal widths.

16. A core on a system on a chip having an interface, wherein the interface contains a plurality of interface signal carriers that are configurable, at compilation, such that a first interface signal carrier of the plurality of interface signal carriers is configurable to support different levels of functionality for the interface.

17. The core as set forth in claim 16, wherein a signal carrier width of the first interface signal carrier is also configurable to support different signal widths.

18. The core as set forth in claim 16, wherein the first interface signal carrier is configurable, at compilation, such that the first interface signal carrier is selectively physically present in the interface or not physically present.

19. A method for generating at compilation a core interface for a system on a chip to enable re-use of the core with a different interface configuration, the method comprising:

providing configurable source code representative of the core interface for the system on a chip and identifying parameters of the core interface;

defining configuration parameters of the core interface; and

generating the core interface for the system on a chip from the configurable source code representative of the core interface and the identified parameters of the core interface configurable in accordance with the defined configuration parameters of the core interface.

20. The method as set forth in claim 19, wherein at least one of the configuration parameters of the core interface is defining whether a first interface signal carrier will be physically present in the core interface or not physically present.

21. The method as set forth in claim 19, wherein at least one of the configuration parameters of the core interface is defining different levels of functionality that the core interface supports through a plurality of signal interface carriers.

22. The method as set forth in claim 19, wherein at least one of the configuration parameters of the core interface is defining a signal width of a first interface signal carrier.

Sonics, Inc.
2440 W. El Camino Real, Suite 620
Mountain View, California 94040

Telephone: (650)938-2500
Fax: (650)938-2577

Invoice

DATE	INVOICE #
	30

BILL TO	SHIP TO
VxTel, Inc. Attn: Sanjeev Renjen, PhD. 47697 Westinghouse Drive Suite 100 Fremont, CA 94539	VxTel, Inc. Attn: Sanjeev Renjen, PhD. 47697 Westinghouse Drive Suite 100 Fremont, CA 94539

P.O. NO.	TERMS
51805	Net 30

DESCRIPTION	AMOUNT
Sonics Integration Architecture 2.1	.00
Sonics FastForward Suite 2.1	00
Sonics FastForward Annual Maintenance	00

Thank you for your business.

Total $

VxTel, Inc.

47697 Westinghouse Drive
Suite 100
Fremont, CA 94539
Tel: (510) 979-2100 Fax: (510) 979-2107

Purchase Order

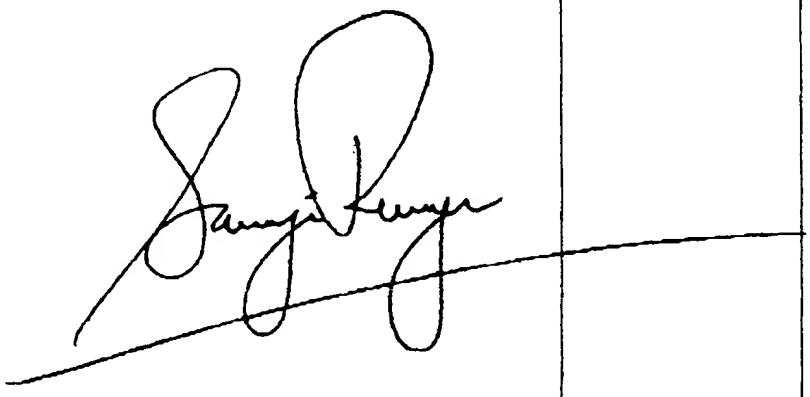
DATE	P.O. NO.
	51805

Vendor
Sonics, Inc. 2440 West El Camino Real, Suite 620 Mountain View, CA 94040

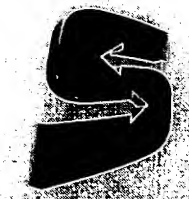
SHIP TO
George Moussa 984-2 Alpine Terrace Sunnyvale, CA 94086

QUOTE NO.	TERMS	SHIP DATE	SHIP VIA	CONTACT
VX122399-TY	Net 30 Days		Electronic	Sanjeev Rensen

DESCRIPTION	QTY	RATE	AMOUNT
Sonics1A - Integrated Architecture Core License - Release 2.0	1	00	.00
Sonics SOC Integration Suite - Annual Subscription price includes one license each of Sosnics SOCIntegrator, SOC Builder and Core Creator	1	.00	.00
Sonics SOC Integration Suite - Annual Maintenance Sent Via Electronic Transmission	1	.00	.00



	Total	$



FastForward™ SOC Integration System

SOC Integration Suite

SOCIntegrator™ and SOCBuilder™ User Reference

Product Version: 2.0

SONICS, INC.

Fast*Forward*™ SOC Integration System

SOC Integration Suite

SOCIntegrator and SOCBuilder User Reference

Table of Contents

Package Core

The final step of *CoreCreator* is the packaging of the core. In this step, the various representations of the core (such as RTL or gate-level netlist), synthesis scripts and configuration files (if applicable), test benches, and documentation, are collected together into a single package for convenient delivery to the system integrator.

The items to be included in the core package are specified using a package file. This file needs to be set up using a text editor, no matter whether the GUI or command line interface is used. In the GUI, there is a separate flow button for packaging the core. Refer to the *CoreCreator User Reference* for a description of this flow function. From the command line, core packaging is achieved using the **corepackage** utility. The **corepackage** utility is described in the *CoreCreator User Reference* .

SOCIntegrator and SOCBuilder

This section describes the steps required to go from a set of packaged cores to a completely integrated SOC as represented by a mapped netlist. Refer to Figure 19 on page 28.

Import Core

Packaged cores must be unpackaged to bring them into the *SOCIntegrator* environment. In the *SOCCreator* GUI, there is an Import Core function to achieve this. From the command line, simply use the Unix **tar** command to extract the cores that were packaged in *CoreCreator*.

Connect & Configure

The SOC is assembled out of a set of cores and a SiliconBackplane. The cores are instantiated and plugged into the SiliconBackplane. Next, the SiliconBackplane and its agents are configured. At this stage the basic SiliconBackplane parameters (address map, bandwidth allocation, flag configuration) and many other agent options are configured. The goal of this step is to create a fully-connected SOC that satisfies the overall system requirements. Requirements such as performance, protocol adherence, chip area, and timing are tested via simulation and synthesis, and adjustments are made to the configuration as needed.

In the GUI, the SiliconBackplane and the cores are instantiated by selecting them from the core catalog. The cores are connected to the SiliconBackplane in the design window. The design window is also used for configuring the SiliconBackplane and its agents, by double-clicking on the entity to be configured. The operations available from the *SOCCreator* design window are described in this document.

From the command line interface, connection and configuration are achieved by describing the SOC in the RTL configuration file. The format of this file is described in the *SonicsIA Hardware Reference*.

Create Netlist

Once the connection and configuration of the SOC has been described, a netlist can be created. In this step, the agents making up the SiliconBackplane are created, each according to the requirements imposed by their configuration, the attached core, and the system. In addition, the SiliconBackplane agents and cores are all wired together to form the chip.

In order to prepare the system for simulation, a system-level test bench is also needed, which at a minimum provides a clock and reset signal to the chip. A sample minimum system test bench is included in the *SonicsIA Hardware Reference*. More typically, there are additional files to describe other pieces of the chip (such as the I/O pad ring) and other pieces of the system-level test bench (such as DRAM models).

In the GUI, there is a dedicated flow button for netlist creation. From the command line interface, the **soccomp** tool accomplishes this step. The **soccomp** command is described in Chapter 6.

Simulation

Now the system is ready to be simulated. As in *CoreCreator*, simulation is divided into three steps: prepare simulation, simulate, and analyze results. Once again, there is a test bench section in the chip RTL configuration file that associates a method with each chip core instantiation for each step of simulation. There is also a test file that specifies which methods to run and what arguments (specifying the test programs or analysis options, for example) to use for a given simulation run. See the *SonicsIA Hardware Reference* for more details about the test bench, as well as a sample test file.

In the GUI, the test bench section is set up using the Define Test Bench flow button. The test file is created using a text editor and selected using the select test file flow button. Once these set-up steps have been accomplished, the three steps of simulation are each kicked off with their individual flow buttons. Refer to Chapter 4 for a description of these flow functions.

From the command line interface, the test bench section is inserted into the chip RTL configuration file using a text editor. Similarly, the test file is created using a text editor. The *SonicsIA Hardware Reference* describes the format of these files. The three steps of simulation are accomplished with the **simprepare**, **simrun**, and **simanalyze** commands. Alternatively, all three can be launched in sequence with the **simall** command. Refer to the tool descriptions in Chapter 6 for more information about these commands.

Synthesize

Logic synthesis is the primary function of *SOCBuilder*. During synthesis, the core and agent RTL files as well as the SOC netlist created earlier are mapped to a specific cell library using a commercial synthesis tool. The output of this step is a mapped netlist that is combined with additional pieces, clocking, test and pad structures to complete the SOC chip design.

In order to run synthesis, a number of items must first be set up. Global parameters such as clock constraints and technology-dependent parameters are specified in the chip synthesis configuration file. Technology-dependent parameters can be extracted from the target cell library using the Technology Compiler tool, **techcomp**, from the command line. For each core, there are core synthesis constraint files to describe the core's timing constraints and a synthesis script to guide the core through synthesis. Ideally, these files are delivered with each packaged core, but if missing they can be entered at this point. Also, any constraints supplied with the core synthesis configuration files can be overridden from the chip synthesis configuration file. Refer to the *SonicsIA Hardware Reference* for an example of a chip synthesis configuration file, as well as for the syntax of that file. See Chapter 6 for more information about the parameters of the Technology Compiler, **techcomp**.

In the GUI, the chip synthesis configuration file is created and updated by supplying the required technology and clocking constraints. The technology information is added into the file using the Edit > Technology menu item, and clocking information is added using the Edit > Clocks menu item. Port constraints, the default values of which come from the corresponding core synthesis configuration files, can be modified through the configuration panes of the cores and agents. Once setup is complete, synthesis is launched using the Synthesize flow button; see Chapter 4.

From the command line interface, technology information, clocking constraints, as well as port constraint overrides are specified by using a text editor to create a chip synthesis configuration file. The format of this file is specified in the *SonicsIA Hardware Reference*. Synthesis is then run using **socmap**. Refer to Chapter 6 for more information about the **socmap** utility.

5 Configuring Your Design

The following sections give a brief overview of each GUI configuration dialog. Complete, parameter-specific configuration information can be found in the *SonicsIA Configuration Guide* and the *SonicsIA Hardware Reference*.

As described on page 12, there are several ways to access the configuration dialog of *any* configurable object, whether it is a SiliconBackplane, agent, core, connection, or monitor:

- Select the object to configure and choose Edit > Configure.
- Double-click on the configurable object.
- Place the mouse pointer over the configurable object, right-click, and choose Configure from the Shortcut Menu.

You may also configure your system by manually editing RTL configuration files (*rtl.conf) and synthesis constraint files (*syn.conf). To do so correctly, however, requires a detailed understanding of SonicsIA and the syntax of these configuration files.

Dialog Basics

Options in dialogs and panes appear grayed out when they are not applicable. They are dynamically enabled or disabled depending upon your configuration.

The number and type of configuration options available depend on the object being configured. Some instances have a large number of configurable options while others have a small number in comparison.

The term "dialog" refers to any query window that appears, asking you to enter or verify information. Sometimes it contains more information than fits in one window so the dialog is divided into tabbed panes. A "pane" describes the information under one tab.

Buttons labeled OK and Cancel appear at the bottom of dialog and pane windows. Press OK to save any changes made to a window and exit. Press Cancel to exit without saving changes.

Note that switching from one pane to another on a multi-tabbed dialog saves any changes made to a pane and cannot be undone with the Undo command. To avoid saving changes to a pane, be sure to press Cancel before moving to another pane.

A default name appears whenever you create an object, but this name can be altered in the name field on the configuration dialog. Almost all names are visible from the design window. (The OCP monitor and external interface are the only objects whose names can only be seen through the configuration dialog and not through the design window.)

Balloon Help

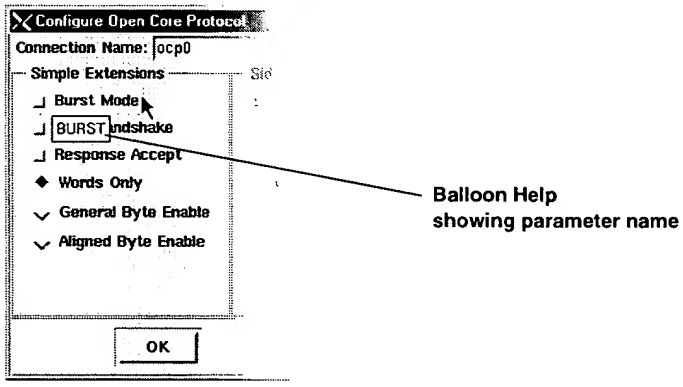
To learn which parameter corresponds to a particular GUI field, move the mouse pointer over the option in the configuration pane and hold the pointer still for a moment. The parameter name appears in the balloon help. Parameter names are used in text files such as the rtl.conf file. They are also useful for identifying and referencing parameters in the *SonicsIA Hardware Reference* or *SonicsIA Configuration Guide*.

The balloon help message on a *locked*, or non-configurable parameter includes the text (Locked by Core) under the parameter name, see Figure 24 for an example.

Special balloon help appears for Clocking panes and for Input Ports and Output Ports panes. When the cursor rests over a value, the valid range for that parameter and attribute appears in a balloon help bubble. Figure 25 contains an example.

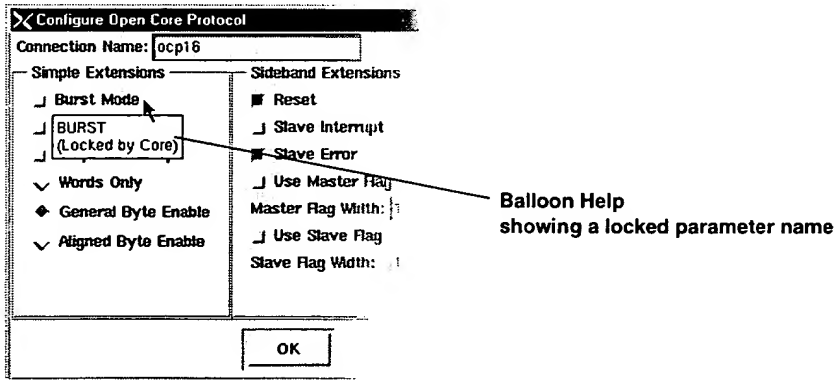
In the following figure, the pointer rests over the OCP burst mode option and the parameter name, BURST, appears in the balloon help.

Figure 23 Sample Balloon Help, Standard



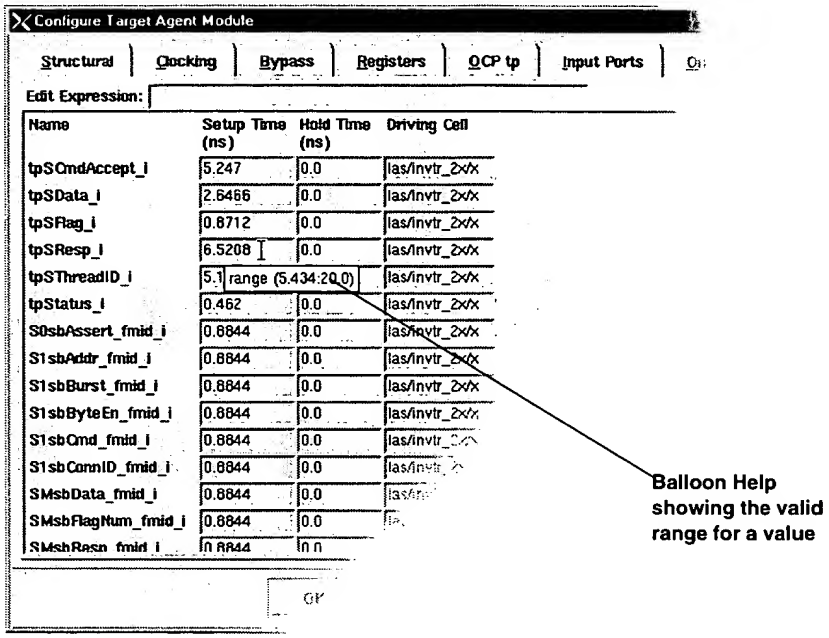
The next figure shows an OCP dialog with locked parameters. The cursor rests over a non-configurable parameter so the balloon help message includes (Locked by Core) under the parameter name, BURST. If you are reading this document in color, notice that the dialog text also appears in a different color (the default is dark blue) to indicate the value is locked.

Figure 24 Sample Balloon Help Showing a Locked Value



In the figure below, the cursor rests over a parameter attribute and the balloon help shows the valid range for the value.

Figure 25 Sample Balloon Help Showing Valid Range



List of Configuration Dialogs

The following lists display the name and page number for each dialog description, both in order of appearance and in alphabetical order.

In Order of Appearance

In Alphabetical Order

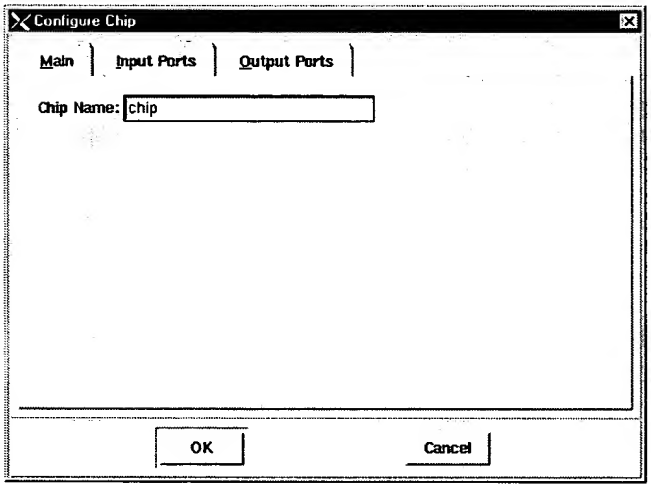
Configure Chip

The Configure Chip dialog allows you to name your design and to specify design-level settings for input and output ports.

Main Pane

As described in Chapter 1, "The Basics", you can rename your chip from the Main Pane of the Configure Chip dialog. There are several different files and filenames associated with a chip. The chip name is the name in the chip command of the rtl.conf file and is used as the name of the top Verilog or VHDL module. This chip name also serves as the default base for the chip filename (rtl.conf file), and for the synthesis constraints file (syn.conf file).

Figure 26 Sample Configure Chip, Main Pane



Input Ports and Output Ports Panes

These values are the port synthesis constraint settings. The size of these panes varies and depends on the complexity of the design.

Input ports panes and output ports panes remain disabled until technology parameters and clock information are defined.

Ports appear on the pane sorted by interface then by port name.

Navigating the Table

The ports pane resembles a spreadsheet: you can navigate the table using the up and down arrows, Shift key + left and right arrows, and the Tab key. The pane, however, does not automatically scroll to follow the cursor so you must use the scroll bar along the right side of the window to page up and down the list.

You can also use keyboard shortcuts to copy and paste values, Ctrl+C and Ctrl+V, respectively. The left and right arrows (without the Shift key) allow you to move through text character by character.

Values and Expressions

Values associated with a port appear in table format to the right of the name. To change a value: click on the value and enter a new one.

To edit a Tcl expression: click on the value, view the expression in the Edit Expression line at the top of the pane, key in any changes to the expression in that line, and press Return. A recalculated value appears.

Figure 27 Sample Configure Chip, Input Ports Pane

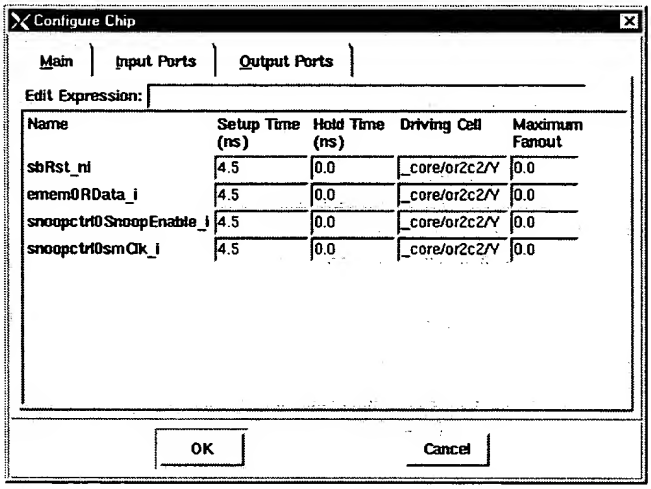
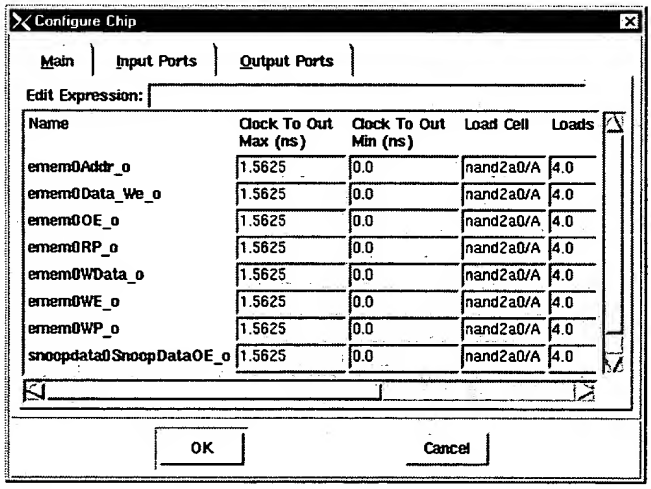


Figure 28 Sample Configure Chip, Output Ports Pane



Configure External Interface

An external interface provides a connection to the chip I/O pads or to parts of the design that are not described in the chip file.

From the configuration dialog for an external interface, you can alter the interface name and interface type, as well as view the bundle type. To change the name of the interface, enter the name in the "Interface Name" field. To change the interface type, click on the upside-down triangle to see the available options for the currently selected external interface and select the appropriate interface type. The interface types available depend on the core (or agent) to which the currently selected external interface is connected. The bundle type is also determined by the core (or agent), but it is not configurable so it appears grayed out.

Figure 29 Sample Configure External Interface Dialog, Example 1

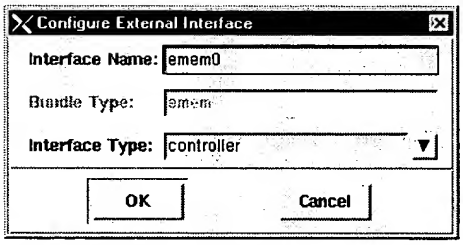
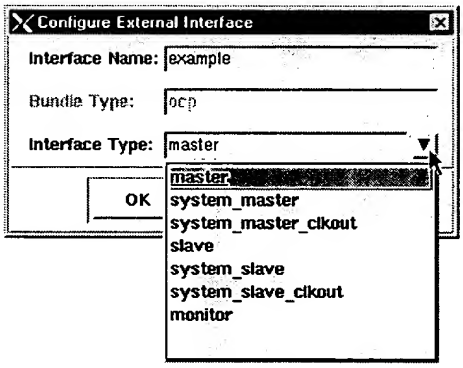


Figure 30 Sample Configure External Interface Dialog, Example 2



Configure Open Core Protocol

The OCP defines a family of point-to-point, bus-independent interfaces using a master-slave transfer model. This dialog allows you to view or define the specific characteristics of an OCP interface or connection. The designer of an individual core may specify a member of the OCP family that matches the capabilities of the core. The family members differ based upon data transfer model, sideband control signaling, and support for concurrency.

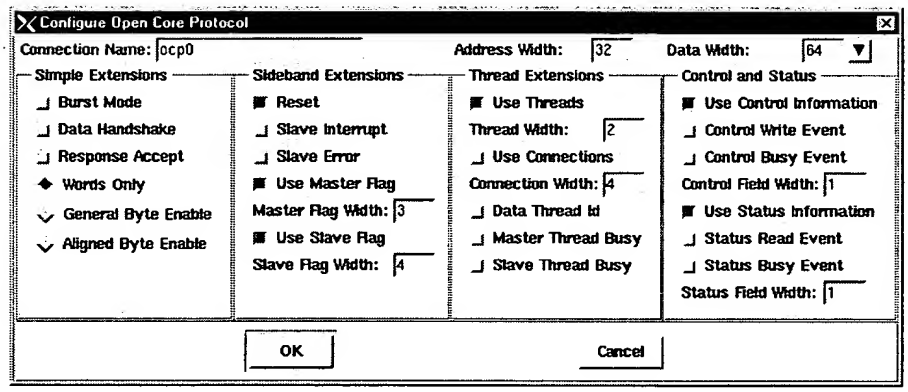
Specific OCP connections *inherit* restrictions from the cores to which they connect. These restrictions frequently prevent you from modifying an OCP parameter that has been pre-determined by an attached core. The full range of family members is available only when both sides of the connection are not restricted. This is typically only true when a flexible behavioral model (such as QCMaster or QCSlave) is attached to a flexible SiliconBackplane agent (such as InitiatorAgent or TargetAgent, respectively).

If any configuration parameters have been inherited from a core, then those parameters appear locked on this dialog. The balloon help message on a locked, or non-configurable parameter includes the text (Locked by Core) under the parameter name, see Figure 24 for an example. The dialog text also appears in a different color than configurable settings (the default "locked" color is dark blue).

You can access OCP settings by configuring the *connection* on the *OCP dialog*. An example dialog appears in Figure 31. You can also access OCP settings by configuring the *core*. This is useful when you have several OCPs on a core and want to modify all the OCP settings at the same time. A core's configuration dialog contains an *OCP pane*[1] for each OCP module interface. An example pane follows in Figure 32 on page 52.

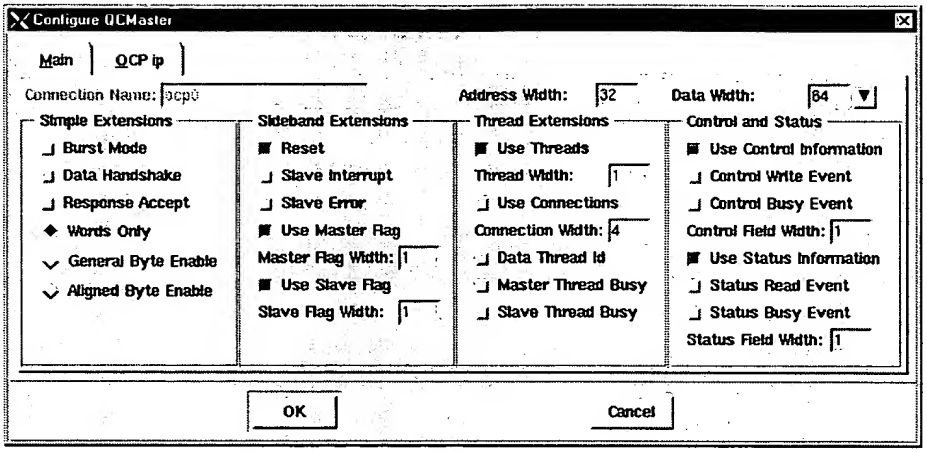
Note that in order to change the connection name for an OCP, you need to change this name on the OCP connection dialog (not on the core dialog or OCP monitor dialog).

Figure 31 Sample Configure Open Core Protocol Dialog



1. The term "dialog" refers to any query window that appears, asking you to enter or verify information. Sometimes it contains more information than fits in one window so the dialog is divided into tabbed panes. A "pane" describes the information under one tab.

Figure 32 Sample Configure Open Core Protocol Pane



For more information about OCP, please refer to the *Open Core Protocol Hardware Reference*. For specifics on parameter configuration, consult the *SonicsIA Configuration Guide*.

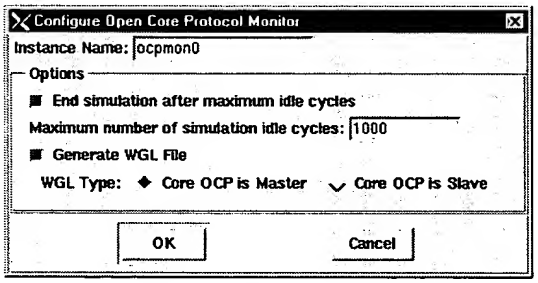
Configure Open Core Protocol Monitor

An OCP monitor is a behavioral module that logs activity on an OCP connection. These logs can be post-processed using the OCP tools.

In addition to logging activity on the OCP, an OCP monitor can be configured to end the simulation when the attached OCP has been idle for a specified number of cycles. To force such an exit, generally speaking, only one OCP monitor in the design needs to be configured to do so.

The OCP monitor can also generate a Waveform Generation Language (WGL) file for later use in a manufacturing test environment.

Figure 33 Sample Configure Open Core Protocol Monitor Dialog



Introduction to QC and QS Models

The QC and QS models are behavioral models that you can use in your design to model cores or to model the rest of the system.

QC stands for Quick Core; QC models are used in *SOCCreator*. A Quick Core Master, QCMaster, models a master core. Similarly, a Quick Core Slave, QCSlave, models a slave core.

QS stands for Quick System; QS models are used in *CoreCreator* (but are also available in *SOCCreator*). A Quick System Master, QSMaster, models the rest of the system to a slave core. And similarly, a Quick System Slave, QSSlave, models the rest of the system to a master core.

Figure 34 QCMaster and QCSlave in SOCCreator

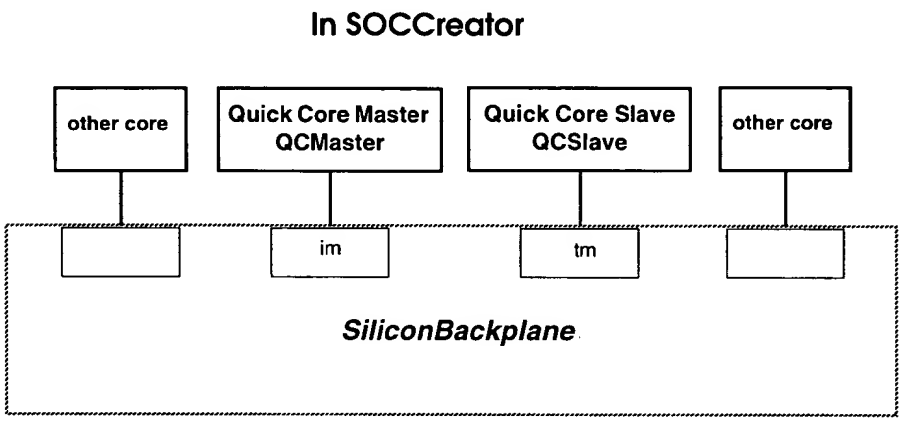
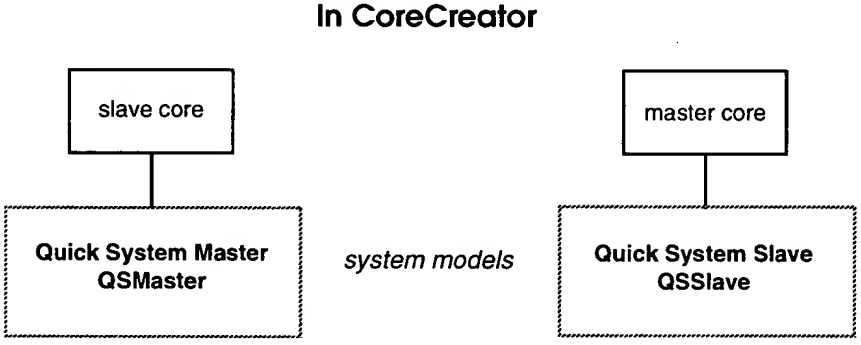


Figure 35 QSMaster and QSSlave in CoreCreator



OCP Pane

OCP panes only appear in instances with configurable OCP interfaces. The label that appears at the top of an OCP pane is the instance name of the port that the pane configures.

For more information, refer to "Configure Open Core Protocol" on page 51.

Input Ports and Output Ports Panes

These values are the port synthesis constraint settings. The size of these panes varies and depends on the complexity of the design.

Input ports panes and output ports panes remain disabled until technology parameters and clock information are defined.

Ports appear on the pane sorted by interface then by port name; ip come first, then tp, followed by the rest of the interfaces in alphabetical order.

Navigating the Table

The ports pane resembles a spreadsheet: you can navigate the table using the up and down arrows, Shift key + left and right arrows, and the Tab key. The pane, however, does not automatically scroll to follow the cursor so you must use the scroll bar along the right side of the window to page up and down the list.

You can also use keyboard shortcuts to copy and paste values, Ctrl+C and Ctrl+V, respectively. The left and right arrows (without the Shift key) allow you to move through text character by character.

Values and Expressions

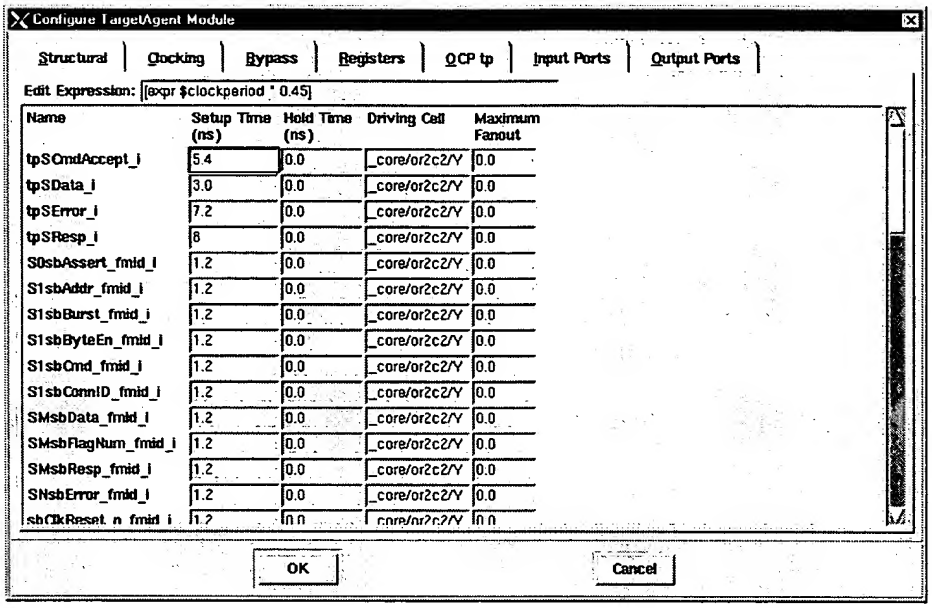
Values associated with a port appear in table format to the right of the name. To change a value: click on the value and enter a new one.

To edit a Tcl expression: click on the value, view the expression in the Edit Expression line at the top of the pane, key in any changes to the expression in that line, and press Return. A recalculated value appears.

For specifics on parameter configuration, please consult the *SonicsIA Configuration Guide.*

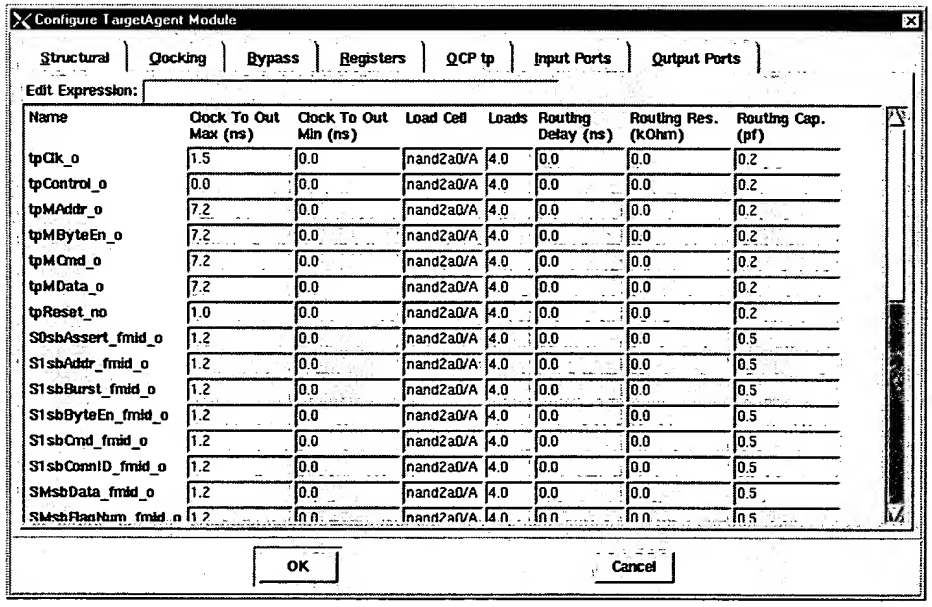
Input Ports Pane

Figure 61 Sample Configure TargetAgent Module, Input Ports Pane



Output Ports Pane

Figure 62 Sample Configure TargetAgent Module, Output Ports Pane



Configure InitiatorAgent Module

The InitiatorAgent Module (IM) is a SiliconBackplane agent that receives OCP requests from an attached master IP core and converts them into legal SiliconBackplane requests. The configuration of an IM's OCP interface is inherited from the characteristics of the attached core. Like the rest of the SiliconBackplane communication system, the configured IM is produced by the **Create Netlist** or **soccomp** command.

Note that many of the IM panes include sections repeated from the TargetAgent Module, since the IM can include an embedded TargetAgent Module to support cores that are both system initiators and system targets (such as DMA engines).

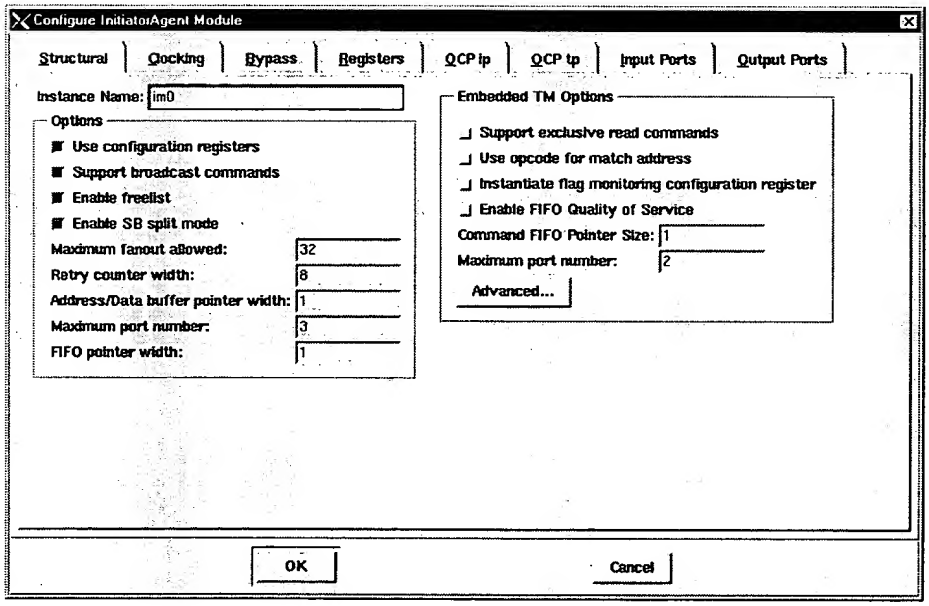
If the IM has an embedded TM, then the TM configuration can be made through this dialog. Otherwise, the embedded TM sections are grayed out because they are not applicable.

Structural Pane

The Structural Pane provides access to FIFO buffer depth, buffer sharing, and other structural parameters of the InitiatorAgent Module. This pane also provides access to TM structural parameters, whenever an IM includes an embedded TM.

For specifics on parameter configuration, please consult the *SonicsIA Configuration Guide*.

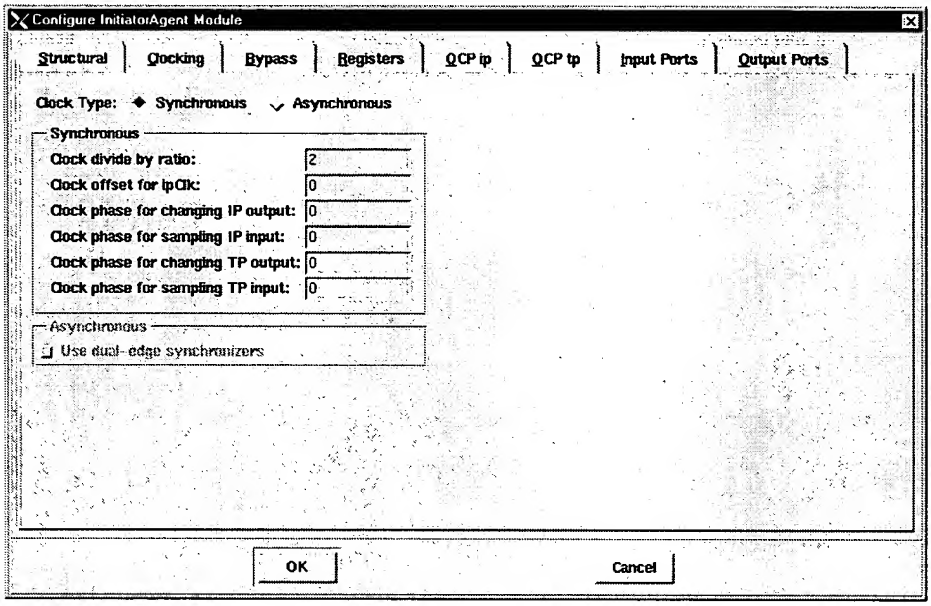
Figure 63 Sample Configure InitiatorAgent Module, Structural Pane



Clocking Pane

The Clocking Pane is very similar to that of the TargetAgent Module. The only extension is to provide separate control over fast IM to slow core timing parameters between the IM's slave and the core's master OCP interfaces.

Figure 64 Sample Configure InitiatorAgent Module, Clocking Pane



<<< Back | Print

Design suite accelerates SOC development

-- 9/16/1999
EDN

You use Sonics' FastForward integration system to develop your system-on-chip (SOC) designs. The system contains intellectual property (IP) that aids multiple-core integration and communication, the Sonics Integration Architecture (SonicsIA), and the Sonics Integration Suite of design tools.

SonicsIA's configurable, modular architecture has a standardized core interface and communication protocol between various cores, such as CPUs, memory blocks, and peripheral blocks. The architecture provides latency and bandwidth service to the cores, along with managing address mapping, control flow, interrupts, and other communication needs.

SonicsIA includes SiliconBackplane, Open Core Protocol, and MultiChip Backplane. SiliconBackplane contains the protocols and interconnect structure to manage interblock communications. SiliconBackplane is a processor-independent, distributed, multiple-master/slave, memory-mapped I/O protocol. It provides both the physical bus and the bus-arbitration functions, merging features such as address-based selection, read and write transfers, and pipelining.

Open Core Protocol is an open standard defining a generic-core "bus wrapper," letting you develop cores independently of an on-chip bus configuration. The Protocol is a synchronous read/write protocol with variable widths and flow control connecting cores to the SiliconBackplane. Open Core Protocol makes the bus conform to a core's needs rather than having you develop cores to meet bus specifications. MultiChip Backplane extends the on-chip SiliconBackplane for chip-to-chip communications. This backplane also lets you verify cores at speeds as high as 400 Mbytes/sec.

The Sonics Integration Suite configures SonicsIA for a design's application and generates a top-level synthesized chip netlist for placement and routing. According to Sonics, an SOC employing SonicsIA and designed with the Integration Suite usually needs only one place-and-route operation. The Suite contains the SOCIntegrator for structural design and functional simulation, and SOCBuilder for driving logic-synthesis, timing-analysis, and test-coverage design tools. The Integration Suite also includes CoreCreator for developing, validating, and preparing cores for reuse. CoreCreator helps develop core representations, including instruction-set simulator and behavioral models, RTL description, testbench, synthesis scripts, and test vectors, that you need during various parts of your design.

FastForward, with SonicsIA and Sonics Integration Suite, runs on Unix platforms. The Sonics Integration Suite has a $64,500 annual license fee. FastForward license fees vary, depending on included core and tool license fees.

Sonics, 1-650-938-2500, www.sonicsinc.com .

—by Jim Lipman

<<< Back | Print

Sonics spins SoC plug-and-play tool

By Michael Santarini, EE Times
September 07, 1999 (1:25 PM EDT)
URL: http://www.eetimes.com/article/showArticle.jhtml?articleId=18302778

MOUNTAIN VIEW, Calif. — Startup Sonics Inc. has announced a new breed of technology that it says enables the plug-and-play of silicon intellectual-property while giving designers greater visibility into a design's interconnect.

The company's FastForward SoC Rapid Integration System promises to allow designers to integrate multiple cores and logic blocks quickly on one chip via a backplane, sidestepping the use of multiple chip buses. The system also automatically controls interconnect in the design, allowing for detailed verification of system-on-chip designs.

Gary Smith, chief EDA analyst at research firm Dataquest Inc., called the tool "a breakthrough technology" that goes beyond plug-and-play.

"It's in a completely new category," said Smith. "I'm trying to figure out a name for the category now, but the best I've come up with so far is behavioral interconnect compiler.

"The tool has the plug-and-play feature of a platform system, but it is much more. It automates the interconnect process. It slices and dices the verification process and makes it attackable."

James Fleury, senior vice president of sales and marketing at Sonics, said FastForward is targeted at simplifying and speeding the overall integration process. One way it does that is by eliminating the restrictions of microprocessor buses on the overall SoC process, Fleury said.

"SoC is largely CPU-centric," he said. "Typically, designers pick a CPU for their design and are stuck conforming the rest of their design to the CPU's particular bus standard or are stuck creating custom interfaces to connect subsystems. What that does is create a maze of wires in a design that is largely unmanageable."

Fleury said FastForward reduces SoC complexity, increases predictability and cuts SoC development time by 50 percent.

The system is chiefly composed of the SonicsIA (Integration Architecture) and Sonics Integration Tool Suite. SonicsIA has three main components: the SiliconBackplane, MultiChip Backplane and Open Core Protocol (OCP). Likewise, the Sonics Integration Tool Suite has three tools: CoreCreator, SOCIntegrator and SOC Builder.

Fleury said SonicsIA's SiliconBackplane provides the protocols and interconnect structure required to manage interblock interactions. He called the component a fully distributed multimaster and -slave, memory-mapped I/O, and processor-independent communication protocol for on-chip, point-to-point interconnect.

The SiliconBackplane merges computing's address-based selection, write and read transfers and pipelining with such communications capabilities as bandwidth decoupling, latency and sideband signaling. Fleury said the component makes the SoC integration process very

observable for validation, test and debugging. He said that all the blocks in the system communicate through the SiliconBackplane and that each block can be tested standalone or in full SoC context.

SonicsIA's MultiChip Backplane extends the SiliconBackplane protocol and provides chip-to-chip communications, allowing system partitioning and hardware prototyping with FPGAs and software development.

The third component of SonicsIA, OCP, is an open standard that defines a generic bus wrapper for cores. Sonics offered an early spec of OCP to the Virtual Socket Interface (VSI) alliance, which used the technology for the transaction protocol portion of its on-chip bus model.

According to Fleury, OCP is a synchronous read and write protocol with variable widths and flow control that connects blocks to the SiliconBackplane.

In the Sonics scheme, designers use CoreCreator to put an Open Core Protocol wrapper around blocks of custom logic or any core, hard or soft, so that it will talk to the SiliconBackplane. CoreCreator can also be used to produce a set of core representations, including instruction-set simulator and behavioral models, RTL, testbench, synthesis scripts and test vectors.

More cores

Designers would then use SOCIntegrator to add the "componentized" core and other "integration ready" cores to SiliconBackplane agents on the SiliconBackplane.

Next, designers would use SOCIntegrator to program the SiliconBackplane. Underlying the tool suite is a technology, called the SiliconBackplane Compiler, that Fleury said generates the connections between blocks based on user-programmable parameters. Designers specify bandwidth, latency, FIFO depth, address/data-bus width, control signals and more.

Designers could also perform functional simulation of blocks or whole designs at this stage. "Users simply specify which parts of the design they want simulated," said Fleury.

Designers would then feed cell libraries to SOCBuilder and use SOCBuilder to guide the SiliconBackplane and componentized cores through Synopsys Design Compiler synthesis. In conjunction with synthesis, SOCBuilder produces the SiliconBackplane in RTL Verilog. Fleury said an upcoming version of the tool will create a VHDL SiliconBackplane.

SOCBuilder also provides inputs for Synopsys' PrimeTime static timing analysis tool at the block and full-chip levels as well as pre- and post-layout. Further, it drives scan insertion for the SiliconBackplane and hooks up core test harnesses.

NEC Corp. is a beta customer of the system. Kojiro Watanabe, senior vice president and general manager of NEC USA, said the tool provided NEC with a direct path from system-level design to VLSI implementation. "Within six weeks, the entire system-chip's performance had been modeled and the SiliconBackplane configuration defined," he said.

In addition, Watanabe said, no changes were required to the system architecture during logical or physical design.

Doug Fairbairn, an industry veteran and management consultant who served a stint as VSI Alliance president and now is a member of Sonics' technical advisory board, said the company "is filling all the cracks between what everyone has been working on in this area. I think FastForward is a true leap ahead for SoC design and the closest thing to plug-and-play that we have seen in the industry."

Grant Pierce, president and chief executive officer of Sonics, said the company will employ a new business model to match the uniqueness of its offering. The company will draw upfront revenue from licensing the tool suite but will also draw per-part royalties for every silicon implementation of its backplane technology.

The fee for a single Sonics Integration Suite annual license is $46,500 running on Sun Solaris, with pricing and longer-term options available.

CEO Pierce said the 24-person, venture-backed Sonics plans to add 15 employees by year's end and to make an initial public offering within two years.